第 33 卷 第 5 期 计 算 机 工 程 2007 年 3 月
Vol.33 No.5 Computer Engineering March 2007
·工程应用技术与实现· 文章编号：1000—3428(2007)05—0217—03 文献标识码：A 中图分类号：TN915.02

一个基于通信系统支持的并行检查点系统

霍志刚[1]，马 捷[2]，孙凝晖[2]
(1. 中国科学院研究生院，北京 100080；2. 中国科学院计算技术研究所，北京 100080)

摘 要：在大规模机群环境下，检查点和恢复机制是一种必不可少的容错技术。该文提出一种基于机群通信系统的可靠性机制，在不作全局同步的情况下获取通信系统全局状态的方法，并利用该方法实现了一个对应用程序透明的并行检查点系统。该系统通过底层通信系统的支持降低了并行检查点的实现复杂度和执行开销，适用于大规模机群应用。

关键词：机群通信系统；并行检查点；容错技术

A Parallel Checkpointing System Based on Communication System Support

HUO Zhigang[1], MA Jie[2], SUN Ninghui[2]
(1. Graduate School of Chinese Academy of Sciences, Beijing 100080;
2. Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080)

【Abstract】 Checkpointing and recovery systems are growing in importance in large-scale clusters. A non-blocking coordinated checkpointing and recovery system is proposed in which reliable communication mechanisms are used to eliminate the overhead of global synchronization. It is shown that a parallel checkpointing system can benefit from supports embedded in low-level communication systems in its implementation and to improve its performance.

【Key words】 Cluster communication system; Parallel checkpointing; Fault-tolerance

1 概述

随着机群规模的不断扩大，并行检查点和恢复系统日益成为机群操作系统中必不可少的组件之一。当一个并行应用正常运行时，并行检查点和恢复系统定期保存该应用中所有进程的当前状态；当该并行应用所涉及的任何一个结点发生故障时，所有进程就被回滚到故障发生前的一次检查点继续运行。多进程检查点系统一般按发起方式分为协同检查点(Coordinated Checkpointing)、非协同检查点(Uncoordinated Checkpointing)和通信引发的检查点(Communication-induced Checkpointing)。协同检查点具有算法简单、对存储空间需求小、不会产生多米诺现象等特点，是机群环境下并行检查点系统的常用方式。非协同检查点和通信引发的检查点由于检查点存储需求较大、算法复杂等原因而较少用于检查点规模庞大、通信密集的机群并行应用条件下。

分布式系统全局一致状态[1]的获取是一个并行应用的检查点过程的重要步骤。对于协同检查点而言，获取全局一致状态就是要消除中途消息(In-Transit Message)。以往的协同检查点系统，如CoCheck[3]等，基于通信通道的FIFO(先入先出)假设，通过全局同步的全交叉方式传递“驱赶消息”来清空通信通道。如果设参与检查点的进程数为N，则在此过程中共有 $O(N^2)$ 个控制消息需要传递。对于大规模机群应用而言，进程数量常数以千计，这导致控制消息数量极大。更关键的是，各个进程之间的同步性不可避免地会受到所在结点的负载、操作系统的调度、并行应用算法的自身特点等因素的影响而难于保证。这些原因使并行应用运行中的全局同步会产生较长时间的阻塞，即使利用性能优异的机群专用通信系统也无法消除其开销。

为了克服线路干扰等因素产生的通信故障，当今主流的机群通信系统都实现了包括消息应答、超时重发等机制在内的各种可靠通信机制。本文将首先引入一种基于通信系统的可靠性机制来避免检查点过程中全局同步的并行应用全局状态获取方法，随后介绍在此基础上实现的一个对应用透明的并行检查点系统。

2 研究背景

2.1 相关研究

与本文提出的并行检查点系统相关的研究领域主要包括低复杂度的分布式检查点算法[2]和MPI应用的系统级容错支持。低复杂度的分布式检查点算法一般需要应用程序在正常运行时统计和维护各信道的消息收发情况，例如在正常消息中夹带一些控制信息等，这会产生额外的正常运行时的计算和通信开销。MPI应用的系统级容错支持一般需要对MPI标准的实现进行扩充或者修改，如FT-MPI、MPICH-V等。

本文所述系统的特点是利用通信系统中的可靠性机制来消除中途消息的隐患，并通过扩展通信系统的功能来保证当任意数量的通信参与者因已知原因而暂停通信之后通信系统状态的可恢复性。可靠通信机制的开销对任何一个成熟的通信系统而言都无法避免；将通信系统全局状态获取的复杂性隐藏于通信系统之中的做法，则使用户可以灵活选取单机检查点工具来构建并行检查点系统。

基金项目：中科院新一代机群关键技术的研究项目(KGCX2-SW-116)
作者简介：霍志刚(1978－)，男，博士生，主研方向：机群通信系统和容错技术；马 捷，副研究员；孙凝晖，研究员
收稿日期：2006-03-05 **E-mail：**zghuo@ncic.ac.cn

2.2 Myrinet 和 MX

Myrinet 是由 Myricom 公司推出的一种广泛用于中高端机群系统中的高速通信网络，大容量 Clos 交换网和高效的可编程通信协处理器是其主要技术优势。Myrinet 有较高的可靠性，并且提供字节级的流控机制。MX 是该公司新开发的用户级通信系统，其中提供了消息应答、超时重发、发送请求备份等可靠性机制。在本文的测试平台上，从主机方测量的 MX 点到点延迟只有 3.2μs。

3 设计

3.1 总体设计思路

在采用消息应答和超时重发机制的通信系统中，发送方通过接收应答消息来确认每个消息的成功送达。对于未在指定时间内得到应答的消息，发送方必须重新发送。这意味着发送方记录着任何一个尚未得到应答的消息。从通信系统的角度来看，已经发出，但尚未得到应答的消息就是中途消息，而传统意义上的检查点过程中的中途消息则是指发送方的进程已经发送，但是尚未被接收方的进程接收的消息。本文以下所指的中途消息都是相对于通信系统而言的中途消息。在检查点过程中，由发送方主动地保存本地的中途消息比由接收方被动地等待所有接收信道中的驱赶消息要更快捷和灵活，并且其开销不会随系统规模增大而呈指数增长，比较适合于大规模机群应用的检查点过程。

本文介绍的通信系统全局状态获取方法的基本思路就是每个进程在检查点过程中只记录本地通信系统的状态，避免全局协同；在应用恢复过程中重发中途消息并过滤重复消息。以下本文将说明当每个进程收到启动检查点过程的广播之后，仅记录本地通信系统的状态，就可以在恢复通信时继续通信。

在任意时刻，一个通信系统中的所有消息必然处于如下 5 种状态之一：待发送，已发送未收到应答，已发送并收到应答，已接收未发送应答，已接收并发送应答。

(1)待发送

这种消息的发送请求信息，如目的结点、目的端口等，在发送请求队列中，消息数据位于发送缓冲区或者用户缓冲区。只要在检查点过程中完整地保存上述信息，就可以在恢复时继续发送这种消息。

(2)已发送未收到应答

发送方无法判定这种消息是否已被接收，因此在恢复时需要再次发送。如果该消息已被目的结点接收，就会产生重复消息，这可以通过每个链接的消息序号而直接予以消除。

(3)已发送并收到应答

这种消息已经完成了网络收发过程，无须在恢复时处理。

(4)已接收未发送应答

在本文的原型系统中发送方对未收到应答的消息采取重发策略，因此接收方在做检查点时直接丢弃这类消息，无须恢复处理。

(5)已接收并发送应答

为了提高通信性能，接收方在发送应答的同时，会将这种消息传递到用户空间，从而使其脱离通信系统的控制范围。如果这种消息的应答未被接收，在恢复通信时就会产生重复消息，需要通过消息序号来清除。

综上，在采用了消息应答和重发机制的通信系统中，通信系统全局状态的获取过程可以直接记录通信系统的本地状态而避免全局协同的开销。

3.2 MX 通信系统的状态保存

在 MX 通信系统中，一个通信端口的状态包括位于主机方的发送请求队列、发送缓冲区、接收缓冲区、接收事件队列和位于 Myrinet 网卡上的端口控制结构以及发送操作控制块链表。一个消息的当前状态可能位于上述的一个或者多个结构中，因此在获取一个通信端口的状态时，上述结构都要完整地予以保存。

由于 MX 是一个典型的用户级通信系统，其发送请求队列、发送缓冲区、接收缓冲区和接收事件队列都位于用户地址空间，任何一个单进程检查点工具无须任何扩展就可以实现这 4 个队列的保存。但是，MX 中的接收缓冲区和接收事件队列均由 Myrinet 网卡的通信协处理器通过 DMA 方式来填充。通信协处理器的运行完全独立于主机方的进程。当主机方的进程开始检查点过程之后，通信协处理器仍会接收新的消息并填充主机方的相关队列。为防止接收数据的丢失和保持数据完整性，应确保在保存通信协处理器可写的各个主机方结构之前，通信协处理器已停止了对应端口的接收操作。

位于 Myrinet 网卡上的端口控制块是记录一个 MX 端口的当前状态的主要结构，其中包括主机方的接收缓冲区和接收事件队列的当前指针、发送操作控制块链表的头指针等。一个发送操作控制块用于记录一个发送请求的处理状态。在本设计中，这些位于通信网卡上的端口信息是进程检查点的一部分，需要写入进程的检查点映像。

在本文的原型系统中，一个 MX 端口的状态保存操作分为 3 步：(1)检查点工具向 Myrinet 网卡发送冻结指定端口的命令，并保存用户空间的发送请求队列、发送缓冲区；(2)在通信协处理器确认端口已冻结之后，检查点工具保存用户空间的接收缓冲区和接收事件队列，以及网卡上的端口控制块和所有中途消息的发送操作控制块；(3)检查点工具向 Myrinet 网卡发送唤醒指定端口的命令。

3.3 降低中途消息的数量

由一次命令广播启动的检查点过程中，各个通信系统冻结端口的时机必然存在先后差别，这会导致一些消息(包括应答消息)的收发双方处于不同的检查点状态，从而形成中途消息。为了使各个通信系统冻结端口的时机尽可能地同步，本文主要针对以下两种情况，在通信系统中进行了处理。

第 1 种情况是发送方尚未冻结端口，而接收方已经冻结了端口，如图 1 中的消息M_1所示。对于这种情况，本设计提供了一种特殊类型的否定应答消息，使其接收端口在主机方发出的端口冻结命令到达之前就停止处理新的发送请求和重发，但继续接收消息。

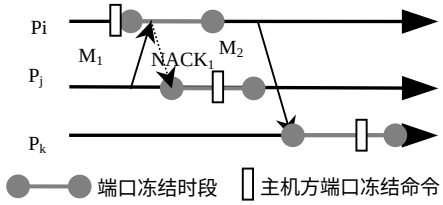

图 1 收发双方检查点状态不同

在保存端口状态的开销很低的情况下，另一种可能出现的情况是一方尚未开始当前一轮的检查点，而另一方已经完成了这一轮的检查点，如图 1 中的消息M_2所示。本设计中为此提供了检查点序号。如果一个消息所携带的检查点序号与接收方的不同，该消息本身或者其应答消息会使检查点序号较小的一方提前进入检查点过程。

4 工程实现

4.1 单进程检查点

由于避免了检查点过程中的全局同步，用户无须修改并

行应用程序或者并行环境库的实现，只要在检查点过程中对每个进程执行单进程检查点操作，其中包括 MX 通信端口的状态保存，就可以实现并行检查点。

本文的原型实现采用了美国劳伦斯伯克利国家实验室开发的 Berkeley Lab Checkpoint/Restart (BLCR)作为单进程检查点工具。BLCR 是一个对用户进程透明的系统级检查点工具。目前 BLCR 尚未对 Sockets、字符设备等特殊文件提供支持，而 MX 通信系统将 Myrinet 网卡以字符设备文件的形式注册到操作系统中，因此本文扩展了 BLCR 对字符设备的支持。如果 BLCR 发现目标进程使用了 Myrinet 字符设备，就会在检查点过程中按照前述的 3 个步骤保存 MX 端口的当前状态。

为了获得更高的通信性能，机群通信系统常常通过异步通信接口使用户进程的执行和网络消息的收发过程实现重叠。在 MX 通信系统中，当一个发送请求或者接收请求被加入对应的请求队列之后，用户进程就立即返回。随后，通信协处理器独立地处理这些请求，而用户进程则通过检测该请求的状态来判断其完成情况。

在本设计中，异步通信的特点被用来降低单进程检查点过程的时间开销。这体现在保存 MX 端口状态的时机被选在 BLCR 执行过程的中段。第一，一个进程在开始检查点过程之后就停止运行，无法提交新的发送请求。如果 MX 系统在这段时间内继续进行消息的发送、接收和应答，在理想情况下会在 MX 端口冻结之前将网络中所有的中途消息处理完毕。第二，在检查点过程结束之前唤醒 MX 端口，可以将恢复通信的时间和检查点过程重叠。

4.2 检查点控制器

基于通信系统可靠性机制的通信状态获取方式使得检查点过程中避免了进程间的全局协同，从而简化了检查点控制器的设计和实现。当得到了一个并行应用中每个进程的所在结点名、进程号之后，检查点控制器启动一次检查点的过程就是向每个进程的所在结点发送对应进程的单进程检查点命令(图 2)。

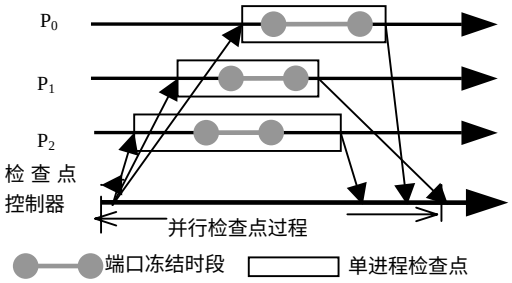

图 2 并行检查点过程

本文的原型系统实现了 MPI 应用的并行检查点。MPICH(MPI 标准的一种比较流行的实现)默认的应用加载器 mpirun 得到了扩充。扩充后的 mpirun 可以接收用户从命令行输入的检查点请求或者定期执行检查点操作。作为 MPI 应用的加载器，mpirun 在 MPI 应用启动时会得到每个进程的所在结点、进程号、MX 网卡号和端口号等信息。当启动一次检查点时，mpirun 就可以直接利用上述信息依次为每个 MPI 进程启动一个子进程，由其在对应的目标结点上执行检查点操作。当执行远程检查点操作的各个子进程都成功返回之后，mpirun 会将当前 MPI 应用中各个进程的进程号、所在结点、检查点映像路径等信息保存在一个检查点控制文件中。

在一个 MPI 应用的恢复过程中，mpirun 首先读取其检查点控制文件，然后在各个指定的结点上执行对应的检查点映像的恢复操作 cr_restart。

5 实验结果

本文的测试平台的配置如下：结点硬件平台为 2 路 1.6GHz AMD Opteron 处理器，2GB 内存；网卡为 Myricom PCIXD-2，其通信协处理器和存储器的主频均为 225MHz。结点操作系统核心为 Linux-2.6.12；MPICH 的版本为 1.2.6；MX 的版本为 1.1.0。

本文采用的测试程序在结点间进行不间断的消息传递。为了提高通信的密集程度，发送方利用异步发送接口每次连续发送 32 个消息。实验中获得的性能数据如下：测试程序的单进程检查点映像的大小约为 880KB，每次并行检查点过程的时间长度约为 80ms。在保存 MX 端口状态的过程中，阻塞式的端口冻结和端口唤醒过程的开销分别约为 16.0μs 和 8.6μs，将网卡上的端口状态保存在检查点映像中的开销约为 54μs (图 3)。MX 端口冻结和唤醒开销的波动很小的原因可以归结为检查点过程和 MX 端口通信过程的有效重叠。在实验中，主机方进程收到检查点的启动信号与 MX 端口开始冻结这两个时间相隔 50ms 以上。在如此长的时间段内，网络中的中途消息已经处理完毕，从而使 MX 端口冻结和唤醒过程比较短。

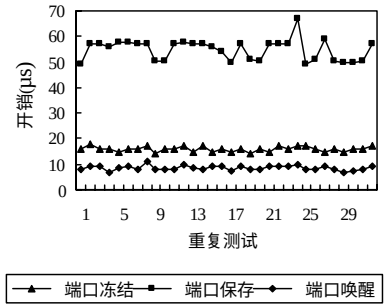

图 3 端口状态保存的开销

6 结论和下一步工作

本文提出了一种无全局同步，快速获取通信系统全局状态的方法。该方法利用通信系统中的可靠性机制，消除了并行检查点过程中的信道清空操作，并实现了检查点过程和通信过程的重叠。基于该方法而实现的并行检查点系统的可行性在实验中得到了有效验证，并且其良好的可扩展性使其适用于大规模机群系统中。

下一步，我们将在该方法的基础上探索并行应用的局部修复机制，即当部分进程的所在结点出现故障之后，在恢复这些进程运行的过程中，需要与之通信的进程会被暂停或者被执行检查点，而不与以上这些进程通信的进程则可以不受影响地继续运行。

参考文献

1 Chandy K M, Lamport L. Distributed Snapshots: Determining Global States of Distributed Systems[J]. ACM Trans. on Computer Systems, 1985, 3(1): 63-75.

2 汪东升, 邵明珑. 具有 O(n)消息复杂度的协调检查点设置算法[J]. 软件学报, 2003, 14(1): 43-48.

3 Stellner G. CoCheck: Checkpointing and Process Migration for MPI[C]//Proceedings of the 10th International Parallel Processing Symposium. 1996: 526-531.